M

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/823,105	03/30/2001	Guei-Yuan Lueh	42390P11280	8390

8791 7590 08/06/2004

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030

EXAMINER	
STEELMAN, MARY J	

ART UNIT	PAPER NUMBER
2122	

DATE MAILED: 08/06/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/823,105	LUEH, GUEI-YUAN	
	Examiner	Art Unit	
	Mary J. Steelman	2122	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *13 May 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-38* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-38* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to Amendment dated 13 May 2004. Per Applicant's request, claims 13-15 and 28-30 have been amended. Claims 1-38 are pending.

Drawings

2. In view of the amendments to the Specification, the prior objections to the drawings are hereby withdrawn.

Specification

3. In view of Applicant's remarks, and amendments the prior objections to the Content of the Specification are hereby withdrawn.

4. In multiple locations of the Specification, the word "field" is misspelled as "filed". As an example, see page 4, last paragraph. Correction is requested.

Claim Rejections - 35 USC § 112

5. In view of Applicant's comments, the prior 35 USC 112 second paragraph rejections are hereby withdrawn.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1-38 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Pre Grant Publication 2001/0047510 A1 to Angel et al., in view of "Poor Man's Watchpoints", by Max Copperman and Jeff Thomas (1995).

Per claims 1, 16, and 31, Angel disclosed:

-compiling a function including a byte code sequence having a field byte code that accesses or modifies a field, the compiled function providing a native code and occupying a code space; ([0054], "the compiler accesses the source code and...converts the source code... (compiling a function)", [0091], "it is useful to instrument memory access instructions (function / fields)...monitoring the variables (fields) of a program that access memory...", [0111], "...instructions being instrumented relate to memory variable (byte code that accesses field) accesses...", Abstract, lines 1-2, "Instrumenting a computer program to provide instrumented byte code..." (byte code))

-generating an instrumentation code corresponding to a field watch of the field; ([0125], "automatically editing the executable byte code representation of...methods for <u>generating instrumented byte code</u>." (emphasis added), [0130[, "There are many different ways to instrument byte code...the editing is performed automatically as a separate post-compile process before the byte code is executed (statically)...")

-inserting the instrumentation code to the native code. ([0133], "a user can supplement the byte code provided in the class instance with separate native code that may be used in conjunction with the byte code.") Regarding a method, computer program product, and system, Angel disclosed a method of byte code instrumentation. See FIG. 1-4 showing system and a computer

program product embedded therein. At [0014], "instrumenting a byte code computer program..." Methods are further disclosed on pages 18-19.

Angel failed to disclose enable / disable options associated with the monitoring function added through instrumentation. However, Copperman and Thomas, disclosed "guarding execution of the instrumentation code if the field watch is not activated". On page 40, The Debugger, 2nd paragraph, "When a watchpoint command is entered or enabled..." Copperman and Thomas discussed adding watchpoints through instrumentation, to debug software. They included features to enable / disable (guarding execution) the watchpoints.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Angel's invention to include optional user features such as guarding execution of instrumented code, including watchpoints, because it makes the software user interactive and more flexible when debugging. Note that Angel discloses the use of a GUI [0208], but does not elaborate on user features likely to be available.

Per claims 2, 17 and 32, Copperman and Thomas disclosed:

-executing a field watch sequence. (Page 40, The Debugger, 3rd paragraph, "On receiving a watchpoint command, the debugger has to add an entry to the watch table and ensure that <cmd> (field watch sequence) is executed when the watchpoint is hit.")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Angel's invention to include optional user features such as guarding execution of instrumented code, including executing a field watch sequence, because

executing a related function makes the software user-interactive and more flexible when debugging.

Per claims 3, 18, and 33, Copperman and Thomas disclosed:

-comparing a flag with a predetermined watch value to determine if the field watch is activated. (Bottom of page 38, "A flag passed to the post-loader designates loads, stores, or both as patch targets." (Flag determines watch value to be activated.) As noted in the rejection of claim 1 above, an enable / disable option exists.)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Angel's invention to include optional user features such as guarding execution of instrumented code, including monitoring watchpoints through the use of flags, because it makes the software user interactive and more flexible when debugging. The use of a flag in software is well known.

Per claims 4, 19, and 34, Angel disclosed:

-inserting the instrumentation code before a field access or modification point. ([0143], "entry of the method is instrumented (before a field access point)...")

Per claims 5, 20 and 35, Angel disclosed:

-inserting the instrumentation code at end of the code space. ([0149], "exit point is instrumented.")

Per claims 6, 21, and 36, Angel disclosed:

-updating an offset of a jump instruction to a stub having the field watch sequence when the field watch is activated. ([0176], "The records contain the new offset of the byte code instructions, which are moved due to insertion of instrumentation instructions.", [0181], "the code table…to reflect the new offsets of the instrumented byte code…", [0182], "byte code is modified to update branch (jump) instructions to reflect the new offsets…" Angel disclosed stubs at [0119], "indirect function calls added by instrumentation are set to 'stub' routines…", meaning empty placemarked routines, whereas Applicant infers that the 'stub' is filled in with code containing 'a field watch sequence when the field watch is activated'. Angel disclosed instrumented routines at [0091], "it is useful to instrument memory access instructions (fields)…monitoring (activated) the variables (fields) of a program that access memory…")

Per claims 7, 22, and 37:

Angel provided details regarding the alteration of the code prior to compiling and branching (jumping) to instrumented code. Angel failed to disclose instrumentation code that could be enabled / disabled.

However, Copperman and Thomas disclosed:

-replacing a no-op sequence with a jump instruction to a stub having the field watch sequence when the field watch is activated. (Page 37, Introduction, 4th paragraph, "code patching - replacing each store and/or load instructions with an inline check or call to a function that gives control to the debugger if the accessed location is being watched, and subsequently executes…" Also, page 40, The Debugger, 3rd paragraph, "On receiving a watchpoint command, the debugger

has to add an entry to the watch table and ensure that <cmd> is executed (jump to stub / instrumentation code) when the watchpoint is hit. Also, page 40, last paragraph, "user's command must be executed at the patch target (stub / instrumented code)." Copperman and Thomas, page 41, 3rd paragraph, "added code to...disable, enable (replace no-op with jump) , and cancel individual watchpoints.")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Angel's invention by including information as provided by Copperman and Thomas regarding jumping to an inserted patch / instrumented code portion / stub for the purpose of optimizing through instrumentation, when altering control flow (enabling / disabling code execution) of a program as these are well known techniques, allowing a user to more flexibly debug software.

Per claims 8, 23, and 38, Angel disclosed:

-saving live global state, the live global state corresponding to an active register; ([0169], "...routine is then patched...at runtime, each call..is intercepted...", [0170], "The patch uses an assembly code thunk that includes a small amount of assembly code and a class instance (data structure) that lets the patch code get control (this is done by saving state) before the native code routine starts...", [0174], "The assembly thunk code may put a pointer...into whichever register (corresponding active register, for saving state)...")

-executing an event hook function for an event corresponding to the field watch; ([0170], "patch code get control before the native code routine starts, and also gets control back when the native

code routine exits." (code is executed), also [0137], "The instrumentation program operates in cooperation with the VM runtime system and may take advantage of particular hooks...")

-restoring the live global state. ([0170], "and also gets control back (restore state from registers) when the native code routine exits.")

Per claims 9 and 24, Angel disclosed:

-pushing the live global state onto a stack. ([0153], "parameters that are passed during instrumentation are passed in a conventional fashion using the stack. Thus, the parameters are pushed on to the stack (pushing live global state onto a stack) prior to invocation of the monitoring function being called.")

Per claims 10 and 25, Angel disclosed:

-passing an argument corresponding to the field; ([0093], "pass a variable pointer (passing an argument) to a function and have that pointer (corresponding to the field) be assigned to another variable within the function...", [0114], "The run time instrumentation node may be a function call to a run time instrumentation function that uses the child node as one of the arguments and returns the value of the child node from the function call to make the value available for the operation node.")

-calling a run-time library function related to the event. ([0113], "each of the specific run time instrumentation routines that is provided may include a function that is called to perform the

instrumentation operation…", [0119], "The run time instrumentation code may be implemented by using a separate set of routines (run-time library function) that is linkable to the code being instrumented via the function calls…The initialization routine determines if an executable library corresponding to the run time instrumentation routine is available…")

Per claims 11 and 26, Angel disclosed:

-retrieve the live global state from the stack. ([0153], "parameters that are passed during instrumentation are passed in a conventional fashion using the stack…" It is well known that a VM is a stack machine, pushing and popping variables to / from the stack.)

Per claims 12 and 27:

Angel disclosed adding instrumentation through alteration of the code prior to compiling. Angel failed to disclose enabling / disabling the field watch.

However, Copperman and Thomas disclosed "activating / clearing the field watch by setting the flag" at page 40, Maintaining The Watch Table, 2nd paragraph, "When a command is disabled (cleared) or canceled, the last range in the table is copied over the range that is no longer being watched…If the table is empty, $fp is set to zero…, page 41, 3rd paragraph, "…code to…disable (clear) , enable (activate) and cancel individual watchpoints…"

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Angel, by using Copperman and Thomas's disclosure that provides more information regarding options for activating / clearing the watch field when

instrumenting code because these features allow interactive user control, thereby making optimization through instrumentation more flexible.

Per claims 13 and 28, Angel disclosed:

-the function is a JAVA method. ([0014], "instrumenting a byte code (JAVA) computer program…", [0144], "entry of method is instrumented.")

Per claims 14 and 29, Angel disclosed:

-the field is a JAVA field in a JAVA virtual machine. ([0014], "instrumenting a byte code (JAVA) computer program…", [0147], "byte code is inserted into the method to cause a local line number variable (field) to be set to the new line number when the method runs.")

Per claims 15 and 30, Angel disclosed:

-the event hook function is compatible with a JAVA Virtual Machine Debug Interface (JVMDI). ([0137], "Instrumentation program operates in cooperation with the VM runtime system and may take advantage of particular hooks (a virtual machine debug interface) or calls provided by the vendors of the VM runtime system." A JAVA Virtual Machine Debug Interface, JVMDI, is a debug interface, that is trademarked by Sun Microsystems.)

Response to Arguments

8. Applicant's arguments filed 13 May 2004 have been fully considered but they are not persuasive.

(A) Applicant has argued, in substance, the following:

As Applicant has noted on page 16, 4th paragraph, of Amendment filed 13 May 2004, there is no motivation to combine Angel and Copperman because neither of them addresses the problem of compilation according to a field watch.

Examiner's Response:

In response to applicant's argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988)and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, Angel and Cooperman are justifiably combined as both are related to (Angel, [0003]) "instrumentation of code in order to facilitate debugging." The claims are of course directed to "generating an instrumentation code" during compilation, and as such the references are applicable. Angel failed to disclose various options associated with the monitoring function added through instrumentation. Thus, the Copperman and Thomas reference was combined to provide additional information relating to software debugging. Note that Angel discloses the use of a GUI [0208], but does not elaborate on user features likely to be available.

(B) Applicant has argued, in substance, the following:

As Applicant has noted on page 16, 4th paragraph, of Amendment filed 13 May 2004, the combination of Angel and Copperman "does not disclose, suggest, or render obvious (1) compiling a function including a byte code sequence, (2) generating an instrumentation code corresponding to a field watch, (3) guarding execution of the instrumentation code if the field watch is not activated; and (4) inserting the instrumentation code to the native code."

Examiner's Response:

See response to claim 1 above.

(C) Applicant has argued, in substance, the following:

As Applicant has noted on page 16, 4th paragraph, of Amendment filed 13 May 2004, "neither of them (Angel or Copperman) addresses the problem of compilation according to a field watch."

Examiner's Response:

"Compilation according to a field watch" is not a claim limitation.

(D) Applicant has argued, in substance, the following:

As Applicant has noted on page 16, 5th paragraph, of Amendment filed 13 May 2004, regarding the Angel reference, "There is no teaching or suggestion of the use of a field watch in generating instrumentation code", but rather that Angel teaches "methods of automatically editing the executable byte code representation…for generating instrumented byte code. (Angel: page 11, [0125])"

Examiner's Response:

Editing the executable byte code...for generating instrumented byte code is generating instrumentation code. Additionally, Angel: [0111], "FIG. 10 illustrates an embodiment of the invention where the instructions being instrumented relate to memory variable accesses and scope changes (an instruction being instrumented, i.e. generating instrumentation code, could be a field watch, i.e., related to memory variable accesses). In other embodiments of the invention, it is possible to instrument other types of IR instructions (generate instrumentation code using a field watch), depending upon which instructions are deemed appropriate for monitoring program (monitor through the use of a field watch?) operation at run time. For example, it may be possible to add instrumentation to monitor run time performance of the program. Other examples of possible uses of instrumentation include, but are not limited to, code coverage analysis and run time error handling." Angel does suggest that instrumentation code corresponding to a field watch could be generated.

(E) Applicant has argued, in substance, the following:

As Applicant has noted on page 17, 1st paragraph, of Amendment filed 13 May 2004, "A watchpoint or watchpoint command is not the same as a field watch..."

Examiner's Response:

A web definition found states, "A watchpoint is a type of breakpoint that is triggered whenever the class field being monitored is modified..." Angel suggests generating instrumentation code for the purpose of monitoring. A watch point and field point are closely related.

Additionally the JAVA Debug Interface defines "Interface AccessWatchpointRequest: Request for notification when the contents of a field are accessed in the target VM..." The JAVA Debug Interface defines "InterfaceModificationWatchpointRequest: Request for notification when the contents of a field are accessed in the target VM. When an enabled ModificationWatchpointRequest is satisfied, and event set containing a ModificationWatchpointEvent will be placed on the EventQueue."

Conclusion

9. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

10. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

11. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (703) 305-4564. The examiner can normally be reached Monday through Thursday, from 7:00 A.M. to 5:30 P.M. If

attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on (703) 305-4552.

The fax phone number is (703) 872-9306 for regular communications and for After Final communications. Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Mary Steelman

08/03/2004

ANTONY NGUYEN-BA
PRIMARY EXAMINER